2.1

The JOIN Clause

The Join Clause

- A JOIN clause allows you to access data from two or more tables in a query.
- A join links to tables on a common key between the two tables. Usually the primary key on one table is compared to the foreign key on another table using the equals (=) sign. This is an equijoin or an inner-join. However, other comparison operators are also valid.
- If column names from each table in the join have the same name, they must be qualified with the table name or a table alias.

Below is a basic example of a SQL statement with an inner join clause using **explicit** syntax.

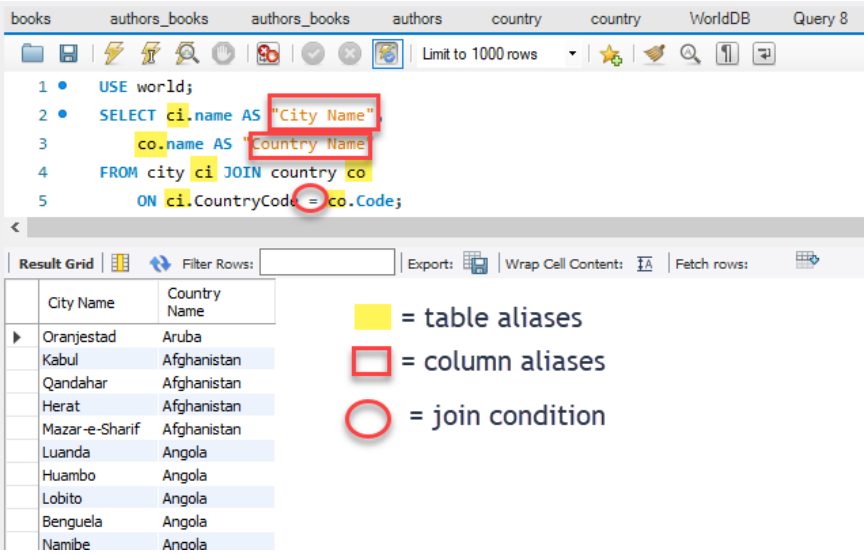
```
1    USE world;
2    SELECT city.name AS "City Name",
3        country.name AS "Country Name"
4    FROM country
6        JOIN city
5            ON city.CountryCode = country. Code;
```

You could write SQL statements more succinctly with an inner join clause using *table aliases*. Instead of writing out the whole table name to qualify a column, you can use a table alias.

```
1    USE world;
2    SELECT ci.name AS "City Name",
3        co.name AS "Country Name"
4    FROM city ci
5        JOIN country co
6            ON ci.CountryCode = co.Code;
```

The results of the join query would yield the same results as shown below whether or not table names are completely written out or are represented with table aliases. The table aliases of co for country and ci for city are defined in the FROM clause and referenced in the SELECT and ON clause:

Results:



Let us break the statement line by line:

USE world;

- The **USE** clause sets the database that we will be querying. You typically have more than one database on your database server. You have to specify which database you are working in.
- The semicolon "**;"** indicates the end of a statement. You can execute multiple statements in sequence by defining each statement with a semicolon

SELECT ci.name AS "City Name", co.name AS "Country Name"

- The **SELECT** clause defines the columns and column order that you want to retrieve in your result set. In this example, we have columns from two separate tables. These columns have the same name, so they MUST be qualified with the full table name or table alias. Otherwise, the column names are ambiguous.
- You separate each column name with a comma "," including the corresponding table alias if one is provided
- To create a friendlier column name in the output, we assign a *column alias* to each qualified column name. Instead of ci.name showing in the column header of the report, we assign a friendlier column alias of "City Name" and for co.name "Country Name."

FROM city ci

- The **FROM** clause specifies the table(s) from which results will be returned.
- In a **JOIN** clause, the first table to be joined is specified after the **FROM** clause.

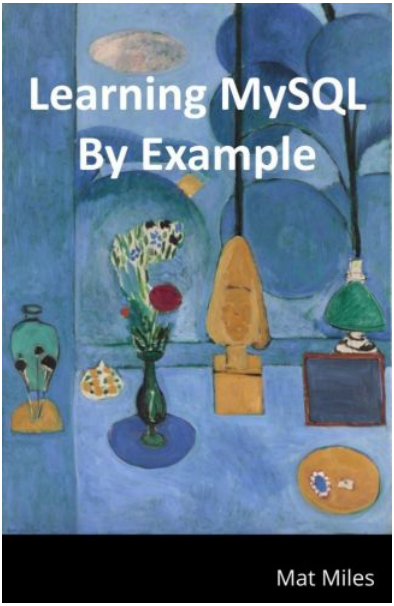
JOIN country co

- Use a **JOIN** clause between the two tables.

- Include the alias if desired.

ON ci.CountryCode = co.Code;

- The **ON** clause specifies the common column from each table (usually a PK in one table and its corresponding foreign key in the other). Each column name is separated with an operator (join condition usually the equals (=) sign.



Miles, M. (2021). *Learning MySQL By Example*. EdTech Books. https://edtechbooks.org/learning_mysql